Tapestry algorithm

COP5615 – Distributed Operating Systems

Submitted By

Sreenivasa Sai Bhasanth Lakkaraju – 41602287 – slakkaraju@ufl.edu

Meghana Reddy Voladri – 43614999 – mvoladri@ufl.edu

Objective:

The goal of this project is to implement the Tapestry Algorithm described in the paper *https://pdos.csail.mit.edu/~strib/docs/tapestry/tapestry_jsac03.pdf*.

Our Approach:

We have implemented the network join and routing as described in the paper. When the user enters the total number of nodes, we are creating 90% of the input nodes at a time and the remaining 10% of the nodes are added to the network using the join function. Once we have the hash value of the new node to be joined, we are comparing it with all the existing nodes in the network and choosing the nodes with largest matching prefix. Once we have the list of the closest nodes, we call these as the neighbor set and the routing tables of the nodes in the neighbor set are updated by adding the new node to be joined (as there is a prefix match and the new node will have an entry in these nodes' routing tables). It will be a better approach to copy the routing tables of one of these neighbor sets as the routing table would almost be same up to some levels (length of the matched prefix). We choose the node that is closest among the neighbor set entries and copy the routing table of that node until the levels matched. Later, we calculate the remaining rows of the routing table.

To run the project,

mix run proj3.exs <numNodes> <numRequests>

Output: Max number of hops taken by all the requests.

Observation:

Input combinations and the maximum number of hops taken are tabulated below.

numNodes	**numRequests**	**maxHops**
100	10	2
1000	10	3
1200	10	4
2000	10	4
3000	10	5

The number of hops taken per combination doesn’t remain constant as the nodes are picked up randomly and there might be cases where the nodes have selected the other nodes which have a good prefix match. In such cases, the nodes might take lesser hops than usual. So, we cannot give an exact estimate of the number of hops taken as it varies with the random node selection.